Proceedings of the 2014 Industrial and Systems Engineering Research Conference
Y. Guan and H. Liao, eds.

The Aircraft Routing with Air Refueling Problem: Exact and Greedy Approaches

Tanya E. Kannon, Sarah G. Nurre, Brian J. Lunday, and Raymond R. Hill
Department of Operational Sciences
Air Force Institute of Technology, WPAFB, OH 45433, USA

Abstract

We consider the problem of finding the least cost route for an aircraft between a predetermined starting and target location while ensuring either enough fuel is on board or acquired en route via refueling for completion of the route. Other considerations of this problem include adherence to regulations for threat and no-fly zones and to regulations for the minimum fuel needed on an aircraft in the case of an emergency or flawed refueling operation. We prove that this problem is *NP*-hard, thereby motivating the potential value for a heuristic for larger, operationally-focused problems. Two solution methods are proposed: (i) a greedy heuristic that examines paths of arcs and (ii) an exact algorithm which is a non-polynomial extension of Dijkstra's shortest path algorithm. These solution methods are computationally tested on a representative set of realistic missions with starting and target locations. The results validate the greedy heuristic as a real-time solution approach which is shown by the speed and accuracy of the solutions found when benchmarked against the exact algorithm and current practice.

Keywords
Routing, Air Refueling, Greedy Heuristic, Exact Dijkstra's Algorithm Extension

1. Introduction
We consider the problem of finding the least cost route from a predetermined starting location to an ending location while ensuring enough fuel is onboard an aircraft to complete the route. In this context, cost could represent many things, such as fuel expended, time, or difficulty of the selected route. We focus on the case in which cost represents the distance of a route, which in turn is a measure of a route's impact on an aircraft via its likelihood for needing maintenance.

The routing of aircraft to minimize cost is important to the military, specifically in a budget-constrained environment. Aircraft are routed to complete various missions, where a mission could be delivering aid to an impacted nation after a natural disaster, such as the Philippines after Typhoon Haiyan [9], or moving cargo and personnel between locations. Reducing the cost needed to complete these missions, or maintain aircraft after mission completion is desirable.

At the start of a route, an aircraft is assumed fully fueled. Fuel can be acquired during the route at an air refueling location utilizing a tanker. This problem can be modeled using a network, where we seek to route an aircraft from a starting node to an ending node through a series of intermediary nodes which represent air refueling locations and waypoint locations to bypass obstacles such as threat air and no-fly zones. When visiting an air refueling node, the aircraft may refuel. When an aircraft chooses to refuel at an air refueling location complications can occur causing the refueling to abort. When a refueling abort occurs, the aircraft must have enough fuel on board to safely divert to a base (i.e., node in the network) with the capabilities to accept it. This contingency is factored into the problem by stipulating the minimum amount of fuel needed on an aircraft when refueling at an air refueling node. With these considerations, we denote this problem the Aircraft Routing with Air Refueling (ARAR) Problem.

Literature Review Dijkstra's shortest path algorithm [1] is related to this problem as it finds the optimal shortest length path from a source node to a sink node. However, this algorithm does not factor in either the fuel needed to traverse this shortest length path or the ability to refuel.

Refueling along a route is considered by Bush [4], Erdoğan and Miller-Hooks [6], and Sundar and Rathinam [10]. Bush [4] considers the problem of routing aircraft through a series of air refueling locations, where the number of

refueling stops is specified but the fuel loaded at each stop is decided. Erdoğan and Miller-Hooks [6] examine a variation of the vehicle routing problem which considers green, or environmentally friendly, vehicles and the needs of these vehicles to refuel when there is a limited refueling infrastructure. Their problem specifically determines a series of routes, where each route is completed by the vehicle using at most one refueling. After the completion of the route, the vehicle returns to the starting location to refuel and then completes its next route until all its customers are visited. Sundar and Rathinam [10] consider the problem of determining the route for Unmanned Aerial Vehicles (UAV) through a series of targets and refueling depots, such that all targets are visited. Their objective seeks to minimize the fuel expended.

Others have examined an aircraft routing problem, with applications in the commercial airline industry, that is fundamentally different from the focus of the paper. Barnhart et al. [3] consider an aircraft routing and fleet assignment problem which seeks to assign aircraft to flight legs and concurrently routes with these aircraft such that all flight legs are completed. Desaulniers et al. [5] consider a similar problem that seeks to identify the daily aircraft schedule and route by specifically determining the time of day for each flight leg and aircraft used. For a survey of the plethora of research relevant to these type of aircraft routing problems, please see Barnhart et al. [2].

We perform a complexity analysis to motivate the solution approaches we take to solve the ARAR problem. We prove that this problem is *NP*-hard, which motivates the development of heuristic solution approaches. The greedy heuristic that we propose takes into account the current location of the aircraft and paths of up to four arcs in the direction of the ending location. This heuristic then greedily selects the shortest cost path from the set of options explored. If an aircraft needs to refuel during the route, the best of the following two options is implemented: (i) directly routing the aircraft to an air refueling node or (ii) altering the determined path to incorporate an air refueling node.

The greedy heuristic is benchmarked against a new exact algorithm which extends Dijkstra's shortest path algorithm [1]. The extension involves redefining the distance label associated with each node to include the fuel upon arrival and whether a refueling operation occurs. In this algorithm each node in the network can accrue a set of more than one distance label, thereby making it a non- polynomial time algorithm.

Computational tests are performed using these solution approaches on a data set representing a realistic set of military missions. Within this data set are starting locations, ending locations, air refueling locations, and appropriate aircraft per mission. The results of these tests demonstrate the effectiveness of the greedy heuristic in both the speed of acquiring a solution and solution quality.

Main Contributions The main contributions of this paper are (i) a classification of the ARAR problem as *NP*-hard, (ii) an exact algorithm which uses Dijkstra's shortest path algorithm as a framework, and (iii) a simple, yet elegant greedy heuristic that is shown to be a high quality solution approach for real-time decision making activities.

The paper proceeds as follows: in Section 2 we formally define the problem; in Section 3 we prove the ARAR problem to be *NP*-hard; exact and greedy solution methods are proposed in Section 4; in Section 5 we examine the performance of the greedy heuristic as compared to the current best practices and exact solution method; and in Section 6 we conclude and propose directions for extending this research.

2. Problem Description

We now present the formal mathematical description of our problem. Given a network $G = (N, A)$ with node set N and arc set A, associate with each arc $(i, j) \in A$ a cost c_{ij} and fuel f_{ij} needed to traverse this arc. Associate with each node $i \in N$ a boolean parameter α_i which equals 1 if an aircraft has the option to refuel at this node. A value m_i associated with each node $i \in N$ represents the fuel needed upon arrival at node i if a refueling operation occurs. This allows the aircraft to make it safely to a nearby node in case of complications during refueling causing a refueling abort. It is true that for all $i \in N$, if $\alpha_i = 0$ then $m_i = 0$.

We seek a route $r = \{a_1, a_2, \ldots, a_{n_r}\}$, from the set of all possible routes R in the network, which is a subsequence of the node set N such that a_1 is the given starting node, a_{n_r} is the given ending node, $a_i \in N$ for $i = 1, \ldots, n$, and $(a_i, a_{i+1}) \in A$ for all $i = 1, \ldots n-1$. Further, we must determine whether or not to refuel at each node a_i for $i = 1, \ldots, n_r$, in route r. Let F represent the fuel capacity of the aircraft utilized. The fuel in the aircraft decreases by f_{ij} when arc (i, j) is traversed. Therefore, with each route $r \in R$, we must ensure that the fuel upon arriving at each node $a_i \in r$ is greater than or equal to zero if no refueling occurs at node a_i, and greater than or equal to m_{a_i} if refueling occurs at a_i. While adhering to these requirements, we seek the route and refueling decisions that minimizes the cost of the route,

specifically

$$\min_{r \in R} \sum_{i=1}^{n_r - 1} c_{a_i a_{i+1}}, \tag{1}$$

which we refer to as the aircraft routing with air refueling (ARAR) problem.

3. Theoretical Results

We next prove that the ARAR problem is *NP*-hard. If the shortest cost path, $P \subset R$, in the network does not require air refueling, specifically $\sum_{(i,j)\in P} f_{ij} \leq F$, then the shortest cost path in the network is the optimal solution where no refueling occurs. Otherwise, if the shortest cost path in a simple path-like network, where each node i only connects to nodes $i+1$, does require air refueling, and it is known that the optimal solution does not require refueling, the knapsack problem [7] can be reduced to an instance of this ARAR problem. Therefore, by showing that the ARAR problem for which it is known that no refueling is required on a simple path-like network (see Figure 1) is *NP*-hard, we can conclude that the general ARAR problem, in which refueling can occur on more complex networks, is also *NP*-hard.

Theorem 3.1. *The aircraft routing with air refueling (ARAR) problem when no refueling is required is* NP*-hard.*[1]

Proof. We reduce the knapsack problem to an instance of the ARAR problem when the optimal solution does not require refueling. Associated with the knapsack problem is a knapsack size W, set of n elements E, size s_i and value v_i for all $i \in E$. The knapsack problem seeks to find a subset $S \subseteq E$ of the n elements such that $\sum_{i\in S} s_i \leq W$ and $\sum_{i\in S} v_i$ is maximized.

Consider an acyclic network for the ARAR problem with $|N| = 2|E| + 2$ nodes and $|A| = 4|E|$ arcs by representing each element $i \in E$ by a pair of nodes $\bar{i}$ and i'. Include a node s which connects to element 1 through arcs $(s, \bar{1})$ and $(s, 1')$ with cost and fuel of zero, and node t which is connected to element $n = |E|$ through arcs $(\bar{n}, t)$ with cost equal to $V - v_n$ and fuel of s_n for an arbitrarily high value $V > max_{i\in E} v_i$, and (n', t) with cost of V and fuel equal to 0. Other arcs are included in the network that connect each node representing element i to each node representing element $i+1$ for $i = 1, \ldots, n-1$ through arcs $(\bar{i}, i\bar{+}1), (\bar{i}, i+1'), (i', i\bar{+}1)$, and $(i', i+1')$ as depicted in Figure 1, where all arcs leaving node $\bar{i}$ have cost equal to $V - v_i$ and fuel equal to s_i, and all arcs leaving node i' have a cost equal to V and fuel value equal to zero.

Therefore, setting the fuel capacity of the aircraft $F = W$, the starting and ending locations to s and t, respectively, if we solve an instance of the ARAR problem on this network we arrive at a solution for the knapsack problem where, if node $\bar{i}$ is visited, it is determined that element i is included in the knapsack and if node i' is visited, it is determined that element i is not included in the knapsack. To traverse the network from s to t we must visit either $\bar{i}$ or i' for each $i = 1, \ldots, n$. The objective function value of the ARAR problem will be $nV - \sum_{i\in S} v_i$ which determines the exact set S of elements selected for the knapsack. □

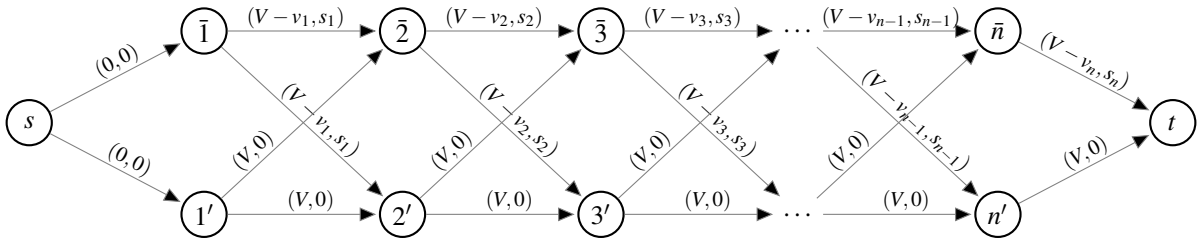

Figure 1: Graphical representation of the knapsack problem as an instance of the ARAR problem

[1] We note, that if f_{ij} and c_{ij} are related, as in for any two arcs (i, j) and (k, ℓ), if $f_{ij} < f_{k\ell}$ then $c_{ij} < c_{k\ell}$ that the problem when no refueling is required is not *NP*-hard and can be solved using Dijkstra's shortest path algorithm with the fuel values as cost. However, we are considering the general case without this assumption.

4. Exact and Greedy Solution Methods

Because of this theoretical result, we propose a greedy heuristic that examines subsets of arcs that form paths from the current aircraft location. We also describe a non-polynomial modification to Dijkstra's shortest path algorithm that arrives at an optimal solution to the ARAR problem for use in benchmarking the proposed heuristic.

4.1 Greedy Heuristic

The greedy heuristic that we propose is simple, yet has promising computational results, as we will demonstrate in Section 5. The idea behind the heuristic is to begin at node s and incrementally add on arc (i, j) to the route when arc (i, j) is (i) the first arc in the shortest 1-, 2-, 3-, or 4-arc path that leads to the ending location, otherwise (ii) the shortest arc from the current location. This greedy approach allows for a detailed assessment with each arc addition of the needs for refueling.

When trying to add arc (i, j) to the route, we determine whether the aircraft can successfully complete this route without additional fuel. If no additional fuel is needed, arc (i, j) is added to the route. Otherwise one of three cases occur. First, if there exists an arc from the current location to a refueling node that can be traversed, the shortest of these arcs is added to the route. Second, if there exists a node in some arc in the current route where refueling is possible but does not currently, occur, we indicate to refuel at the node closest to the current location. Otherwise, the route is modified by removing the last arc from the route and determining if from this new location and fuel level, an arc to a refueling location can be added. This modification of removing arcs from the route continues until the aircraft can be successfully routed to a refueling node. The main step of the heuristic is then repeated until the arc that is added to the route leads to the ending location. Pseudocode for this heuristic is outlined in Algorithm 1.

Algorithm 1 Greedy Heuristic for the ARAR Problem

Input: Network $G = (N, A)$ where nodes $i \in N$ have associated parameters α_i, m_i, and arcs $(i, j) \in A$ has associated fuel f_{ij} and cost c_{ij}.
Input: Starting location s and ending location t.
Set $Target\_Reached =$ FALSE.
Set $Current\_Location = s$.
Set current route $r =$ NULL.
while $Target\_Reached =$ FALSE **do**
    Determine arc (i, j) that is the first arc in the shortest 1, 2, 3, or 4 arc path that leads to t, or the shortest arc $(Current\_Location, k)$.
    if Arc (i, j) can be added to the route r without needing extra fuel **then**
        Add (i, j) to r.
        Set $Current\_Location = j$.
    else
        if $\exists (Current\_Location, k)$ that can be traversed without extra fuel and k is a refueling location. **then**
            Add shortest $(Current\_Location, k)$ to r.
            Set $Current\_Location = k$.
        else if There exists a refueling node i in some arc $(i, j) \in r$ where refueling currently does not occur **then**
            Indicate to refuel at the node i in some arc $(i, j) \in r$ closest to the current location.
        else
            Iteratively remove the last arc from the route, updating fuel and current location, until $\exists (Current\_Location, k)$ that can be traversed without extra fuel and k is a refueling location.
            Add shortest $(Current\_Location, k)$ to r.
            Set $Current\_Location = k$.
        end if
    end if
    if $Current\_Location = t$ **then**
        Set $Target\_Reached =$ TRUE.
    end if
end while
Return r.

4.2 Exact Algorithm

We now discuss an exact algorithm for the ARAR problem that uses Dijkstra's shortest path algorithm [1] as a framework and implements ideas from the A^* algorithm [8] to speed up computation. In Dijkstra's shortest path algorithm, distance labels are assigned to each node in the network to capture the current shortest length to arrive at the node and through which predecessor the route traverses. The node with the smallest distance label is iteratively made permanent, and its neighbors are examined when their distance labels are potentially updated. This process continues until all nodes are made permanent. At the termination of the algorithm, the distance label associated with each node represents the shortest length path from the given starting node. Further, the path to this node can be constructed by tracing back the predecessors. The A^* algorithm improves upon Dijkstra's shortest path algorithm, by adding to the distance label value a heuristic value representing the approximate length of the path from that node to the ending location.

Using both Dijkstra's algorithm and the A^* algorithm as a framework, we extend the idea of applying labels to nodes, but we capture more information with each label and allow each node to acquire multiple distance labels. Let a label ℓ associated with a node be comprised of a septuple of information: (i) distance from the start node $d(\ell)$, (ii) estimate distance to the ending location $e(\ell)$, (iii) fuel upon arrival $f(\ell)$, (iv) a boolean value of whether to refuel or not at this node $\beta(\ell)$, where $\beta(\ell) = 1$ if refueling occurs, (v) the predecessor node $p(\ell)$, (vi) the label number of the predecessor $p\ell(\ell)$, and (vii) label number for this node $n(\ell)$, where (vi) and (vii) are necessary because each node can have multiple labels. The length of the shortest path from each node to the ending location ignoring fuel requirements is used for $e(\ell)$ as this is a lower bound, monotone admissible heuristic function, on the true shortest path when fuel is considered.

At the start of the algorithm, the shortest path from each node $i \in N$ to the ending location is calculated, which we denote SP_i. Using this information, only the starting node i is given a label with values $\ell = (0, SP_i, F, 0, NULL, NULL, 1)$, which assumes the aircraft starts with a full tank of fuel. (This can be easily relaxed by changing F to the starting fuel level in the aircraft.) The algorithm then sets the current node, denoted $c\ell$, to the starting location and makes the only label (ℓ which is associated with the starting node) to be the current permanent label L. The algorithm proceeds by examining all neighbors of the current node, where a neighbor is any node i such that there exists an arc $(c\ell, i)$. A label is added to each neighboring node, where the values of the label depend on whether the neighbor node allows for the option of refueling.

If no refueling occurs at neighbor node i, then the label added has the following values

$$(d(L) + c_{c\ell i}, SP_i, \beta(L)F + (1-\beta(L))f(L) - f_{c\ell i}, 0, c\ell, p\ell(L), k+1), \tag{2}$$

where the distance is calculated as the distance of the current permanent label plus the cost to go from $c\ell$ to i, the estimate distance to the ending location is exactly the shortest path without regards to fuel, the fuel is calculated based on whether refueling occurs at the current label ($\beta(L)$) minus the fuel it takes to traverse $(c\ell, i)$, the refueling indicator is set to 0, the predecessor is set to the current location $c\ell$, the label of the predecessor is calculated based on the label number of L, and the label number for node i is incremented, where k is the current number of labels associated with node i. If refueling is allowed at neighboring node i, then two labels are added to node i: one with the exact same values described when no refueling occurs as presented in Equation 2, and one with a change only in the fourth entry that signifies that refueling occurs, specifically

$$(d(L) + c_{c\ell i}, SP_i, \beta(L)F + (1-\beta(L))f(L) - f_{c\ell i}, 1, c\ell, p\ell(L), k+1). \tag{3}$$

This addition of two labels allows for the option to examine both cases at node i, refueling and not refueling. We note, that a label is not added to a node if the fuel upon arrival (second entry in the label) is less than 0, or less than m_i if refueling occurs. Further, a label is not added if it is associated with a route that includes a cycle.

Emulating Dijkstra's algorithm with A^* additions, we proceed by making permanent not a node, but the label with the current smallest distance plus estimate distance value. The current node is then set to the node for which this label is associated, and the algorithm continues by examining the neighbors of this node. When a label associated with the ending node is made permanent, the algorithm terminates because, at this point, the distance to reach the ending location cannot be decreased. As in Dijkstra's algorithm, the route can be acquired by tracing back the predecessors; however careful attention is given to the predecessor and the label number from which this predecessor comes, as the labels associated with any node i could have different predecessors. Further, when tracing back the path through the

labels, we can determine whether refueling occurs at each node based on the boolean value in the fourth entry of the label which signifies refueling with a value of 1. Pseudocode for this algorithm is outlined in Algorithm 2.

This algorithm is a non-polynomial time algorithm, as the total number of labels that could be acquired in an acyclic directed network is $2^{|N|-1}$. Without loss of generality, in an acyclic directed network we can label the nodes such that the only arcs in the network are (i, j) where $j > i$. Nodes 1 and 2 could have at most one label as there is only one way to reach both of these nodes. There are at most two ways to arrive at node 3, via either node 1 or node 2, therefore making the maximum number of labels at node 3 to be two. For nodes 4, 5, and 6, the maximum number of labels are four, eight, and sixteen, respectively. This process continues where the number of ways to reach node i, or equivalently the maximum number of labels at node i, is exactly the sum of the number of labels for all nodes j when $j < i$, which equals 2^{i-2} for $i = 2, \ldots, |N|$. Therefore, the total number of labels in the network is $1 + \sum_{i=2}^{|N|} 2^{i-2} = 2^{|N|-1}$, demonstrating the non-polynomial nature of the algorithm.

Algorithm 2 Exact Dijkstra's Algorithm Extension for the ARAR Problem

Input: Network $G = (N, A)$ where nodes $i \in N$ have associated parameters α_i, m_i, and arcs $(i, j) \in A$ has associated fuel f_{ij} and cost c_{ij}.
Input: Starting location s and ending location t.
Set *Target_Reached* = FALSE.
Determine the shortest path SP_i from each node $i \in N$ to t.
Set current location $c\ell = s$.
Add label ℓ to s with $\ell = (0, SP_s, F, 0, NULL, NULL, 1)$ and make this label the current permanent label L.
while *Target_Reached* = FALSE **do**
  for all Neighbors i of current location $c\ell$ **do**
    Let k equal the current number of labels associated with node i.
    if Node i is a refueling option i.e., $\alpha_i = 1$ **then**
      Add label $\ell = (d(L) + c_{c\ell i}, SP_i \beta(L)F + (1 - \beta(L))f(L) - f_{c\ell i}, 0, c\ell, p\ell(L), k+1)$ to node i.
      Add label $\ell = (d(L) + c_{c\ell i}, SP_i, \beta(L)F + (1 - \beta(L))f(L) - f_{c\ell i}, 1, c\ell, p\ell(L), k+2)$ to node i.
    else
      Add label $\ell = (d(L) + c_{c\ell i}, SP_i, \beta(L)F + (1 - \beta(L))f(L) - f_{c\ell i}, 0, c\ell, p\ell(L), k+1)$ to node i.
    end if
  end for
  Determine the label with the smallest distance plus estimate distance and set to be the current permanent label L.
  Set current Location $c\ell$ to the node of which L belongs.
  if $c\ell = t$ **then**
    Target_Reached = TRUE.
  end if
end while
Trace back the predecessors to attain route r.
Return r.

5. Computational Results

We compare the greedy heuristic to the exact algorithm and to the current practices used by Air Force Mobility Analysis office. The tests are performed on a realistic data set representing 10 missions, each with provided starting location, target location, and aircraft. This data set involves a cyclic network with 34 nodes and 666 arcs. Each node i is classified to allow air refueling $\alpha_i = 1$ or not $\alpha_i = 0$. Further, for each node that can accept air refueling, a value representing the amount of fuel needed to get to a base (node in the network) in the case of complications is known for each aircraft type.

The current method in practice starts by determining the shortest round trip route with no consideration for fuel expenditure. Based off of this route, the minimum number of times needed to refuel is calculated by dividing the fuel needed for the route by the fuel capacity of the aircraft. Once this number of calculated, adjustments to the route are made by hand that route the aircraft to what appears to be the nearest refueling location by looking at a map. This method is not guaranteed to be exact and can be very time consuming when considering many missions.

All three solution methods are used to a find a round trip route for each mission, which starts at the provided starting location, reaches the target location (ending location), and returns to the starting location. Note that, in some cases, missions require expenditure of fuel while at the target location, which is pre-calculated and provided in the data set. To incorporate the ability for the exact algorithm to find an optimal route trip solution, we duplicate all nodes $i \in N$, denoted i', all arcs in the network $(i, j) \in A$, denoted (i', j'), and add in arc (t, t') for target location t, with cost equal to 0 and fuel equal to the pre-calculated fuel at the target. With this new network, the algorithm seeks to find the shortest path from $s \rightarrow t \rightarrow t' \rightarrow s'$.

For each test we capture the solution and the elapsed time needed to arrive at this solution. The greedy heuristic and exact algorithm were tested using a laptop computer with a 2 GHz Intel Core i7 Processor with 8 GB of RAM, which is representative of the computing resources available to the Air Force Mobility Analysis office. We do note, however, inconsistencies in the software in which the heuristic and exact algorithm were implemented; the greedy heuristic was coded in VBA within Microsoft Excel, and the exact algorithm was coded using C++.

For the current practice method, we were not provided with the individual times needed to find each route, but instead we have the total time needed to solve all 10 instances. Therefore, the reported elapsed times for the current practice are the total time divided by 10. An optimality gap is calculated for both the greedy heuristic and the current practice solution by calculating the percentage difference between the objective function value of the heuristic (or current practice solution) and the objective function value attained via the exact algorithm. The results of these tests are presented in Table 1.

For these 10 tests, we find that both the greedy heuristic and exact algorithm run very quickly in no more than 2 seconds, thereby demonstrating their ability to serve as a real-time solution method. Further, as is demonstrated by the small optimality gaps reported, the greedy heuristic matches or outperforms the current practice method in all but one test instance, and it shows a substantial decrease in the computational time needed to arrive at this solution. Overall, these tests verify that both the greedy heuristic and exact algorithm provide high quality solutions in a tractable amount of time for this application. Further tests should be conducted on a wide range of instances to examine the robustness of both methods for various size networks, as the hardness of the problem indicates that the exact algorithm would not scale well as the network size increases.

	Greedy Heuristic		Current Practice		Exact Algorithm	
Mission	(%) Gap	Time (s)	(%) Gap	Time (s)	(%) Gap	Time (s)
1	2.64%	1.80	0.48%	4320.00	0.00%	0.59
2	1.83%	0.64	10.03%	4320.00	0.00%	1.52
3	11.25%	0.32	31.90%	4320.00	0.00%	0.20
4	2.03%	0.24	2.03%	4320.00	0.00%	0.21
5	0.00%	0.19	0.00%	4320.00	0.00%	0.20
6	0.00%	0.20	0.00%	4320.00	0.00%	0.20
7	0.00%	0.19	0.00%	4320.00	0.00%	0.20
8	0.00%	0.19	0.00%	4320.00	0.00%	0.20
9	0.00%	0.19	0.00%	4320.00	0.00%	0.20
10	3.74%	0.20	3.74%	4320.00	0.00%	0.20

Table 1: Computational results on the set of 10 realistic missions

6. Conclusions

This paper has examined the problem of finding the least cost route between a set of points through a network while adhering to fuel regulations. It was theoretically proven that the problem is *NP*-hard. Two solution methods were proposed: (i) a greedy heuristic and (ii) an exact algorithm which is a non-polynomial extension of Dijkstra's shortest path algorithm. Both methods were computationally tested for a set of 10 missions (provided starting and ending locations) on a realistic network. The results of these tests indicate that both solution methods arrive at high-quality (and optimal in may cases) solutions quickly. This validates both as potential real-time solution methods in practice, as the current practice takes a considerable amount of time to arrive at comparable quality solutions.

Further research should involve examining a thorough set of computational tests on networks of various size to deter-

mine the relative robustness of the solution methods. Additionally, we propose expanding the scope of the problem to include consideration of simultaneously routing more than one aircraft. In practice, the concurrent routing of multiple aircraft occurs and, if not properly managed, can cause congestion and waiting time at air refueling locations. Such an extension would necessarily consider a temporal aspect for the routing as well in order to prevent such congestion. Lastly, we will look to revisit the assumptions within this study regarding the fixed locations of air refueling locations; in reality, decision makers have the ability to change these locations, and it is appropriate to examine this extension as well.

Disclaimer

The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

Acknowledgements

We would like to thank Mr. Donovan at HQ AMC/A9 for his assistance in providing the realistic mission set and insight with regards to the current approach.

References

[1] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B., Network flows: Theory, algorithms, and applications, Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[2] Barnhart, C., Belobaba, P., and Odoni, A.R.,2003, "Applications of Operations Research in the Air Transport Industry," Transportation Science, 37(4), 368-391.

[3] Barnhart, C., Boland, N.L., Clarke, L.W., Johnson, E.L., Nemhauser, G.L., and Shenoi, R.G., 1998, "Flight String Models for Aircraft Fleeting and Routing," Transportation Science, 32(3), 208-220.

[4] Bush, B.A., 2006, "Analysis of fuel consumption for an aircraft deployment with multiple aerial refuelings" Ph.D. dissertation, North Carolina State University.

[5] Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M.M., and Soumis, F., 1997, "Daily Aircraft Routing and Scheduling," Management Science, 43(6), 841-855.

[6] Erdoğan, S. and Miller-Hooks, E., 2012, "A Green Vehicle Routing Problem," Transportation Research Part E: Logistics and Transportation Review, 48(1), 100-114.

[7] Garey, M.R. and Johnson, D.S., 1979, Computers and Intractability: A Guide to the Theory of NP-Completeness, First Edition, W. H. Freeman, San Francisco.

[8] Hart, P.E., Nilsson, N.J., and Raphael, B., 1968, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions of System Science and Cybernetics, 4(2), 100-107.

[9] Miles, D., 2013, U.S. Military to aid typhoon-devastated Philippines, Last Accessed on December 16, 2013: http://www.af.mil/News/ArticleDisplay/tabid/223/Article/467481/us-military-to-aid-typhoon-devastated-philippines.aspx.

[10] Sundar, K. and Rathinam S., 2013, "Algorithms for Routing an Unmanned Aerial Vehicle in the presence of Refueling Depots," Technical Report. Available online at http://arxiv.org/abs/1304.0494.